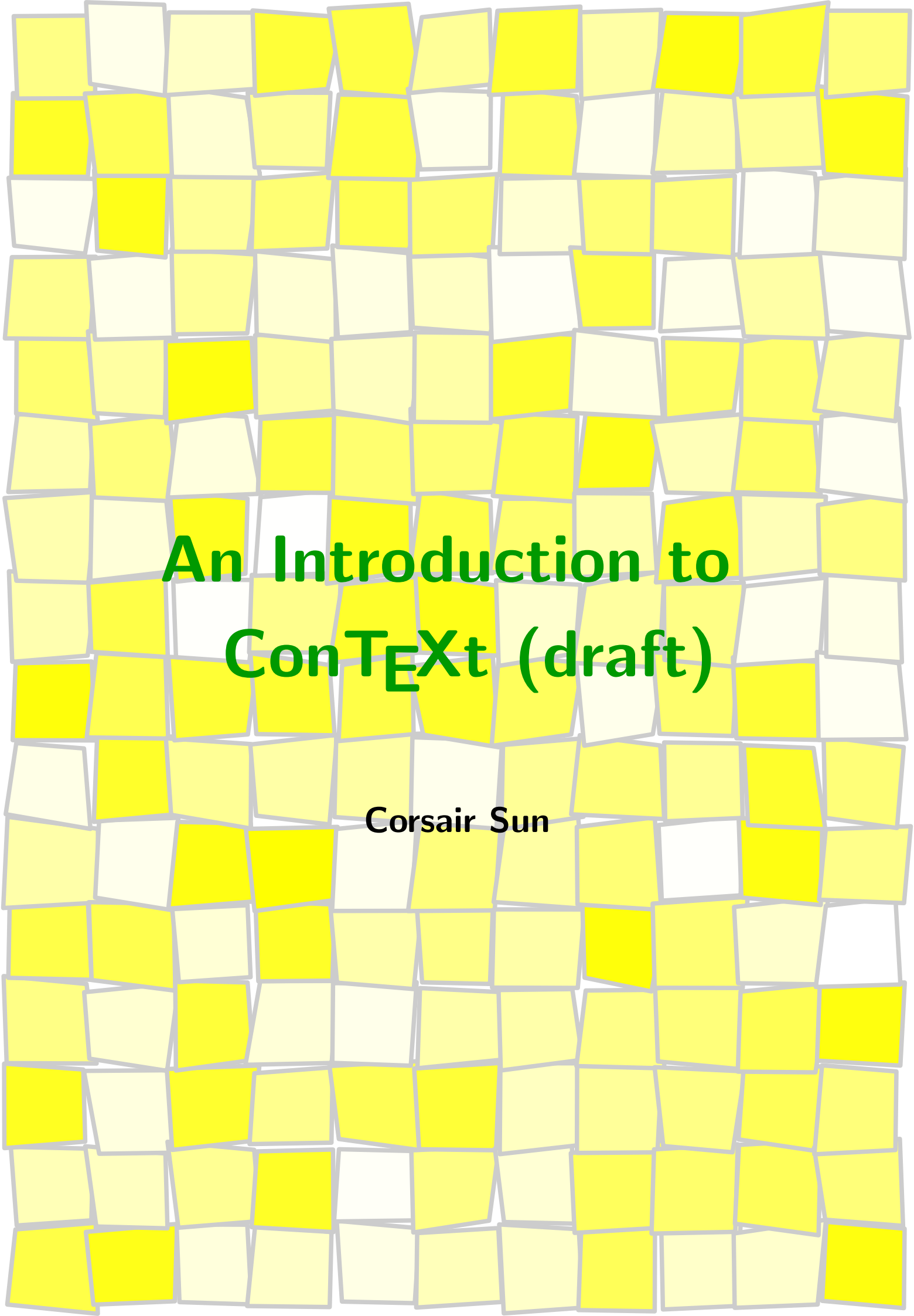


An Introduction to ConTEXt (draft)

Corsair Sun

Abstract

This is an unofficial and very simple manual of ConTEXt powerful typesetting system based on TEX.

Strangely, the official manual (`cont-enp.pdf`) lacks explanations of some general commands such as `\os` and the `table` environment, which are documented separately in either the **ConTEXt wiki**, *This Way* magazine, or other documents. This article tries to accumulate them together in categories, and provides a more readable way to learn the basis of ConTEXt.

Due to this purpose, this article will not cover all aspects of ConTEXt, but some most frequently used features. Also, it will not aim at the technical details of commands. The goal of this article is that after reading, the readers should be able to typeset general-purposed articles and even books.

...

1 Introduction

1.1 Overview

ConTEXt is a typesetting system based one Donald E. Knuth's brilliant TEX. Due to the great algorithm he invented, ConTEXt can generate fantastic-looking documents. At the same time, ConTEXt simplifies the process to customize layouts and styles, and make it possible that config them in a uniformed interface, which looks like

```
\setupsomething[...,...=...]
```

for example, the block above has a background of light gray, which can be archived with

```
\setuptyping
  [before=\startbackground,
    after=\stopbackground]
```

1.2 ConTEXt and LATEX

LATEX is another macro package of TEX. In comparison to LATEX, ConTEXt is, in some way, more low-leveled. Using LATEX, one can hardly meet any plain-TEX commands (except in math mode); however, it's better to have some basic knowledge of plain-TEX when playing with ConTEXt. For instance, to typeset a bold-face word in both TEXand ConTEXt, `\bf` is used. And one may find that, similar to TEX, we use brackets to limit the range that `\bf` applies to like `{\bf word, and another word}`. On the other hand, we use `\textbf{word, and another word}` to do the same thing in LATEX.

From another point of view, ConTEXtis far more high-leveled. In ConTEXt, a whole lots of documents can be maintained as projects, products and element, each of which use a single environment file to have similar style, or separated environment files if we wish. TEX and LATEX lack such a facility. If one wants to maintain a book including some separated files, s/he must use `\input` or `\include` commands, and use some third-party mechanism such as Makefile or AUCTEX if the project is even larger. In addtion, ConTEXt implement a "subpaper" method to adjust page layout, in which we put small papers in bigger papers. Even better, we can have multiple small pages in one large paper.

2 Basic Stuff

2.1 Hello World

2.1.1 First Run

There is no document classes in ConTEXt. To generate a simple document, we use

```
\starttext
Everyone starts with Hello World!
\stoptext
```

Save the file in a name ended with `.tex`, and issue the following in a shell

```
texexec --pdf filename.tex
```

The generated document will be `filename.pdf`.

From this example we know at least two things about ConTEXt:

1. We use a `\starttext` and `\stoptext` pair to brace the text we want to typeset.
2. We use `texexec` to compile the source file.

2.1.2 Some Analyses

We now know that `\starttext` and `\stoptext` are used to brace the main text. In fact, in ConTEXt, most environments starts with `\start...` and ends with `\stop...`. For example, a verbatim environment is run thus

```
\starttyping
You feel a disturbance of \ConTeXt.
\stoptyping
```

We also notice that `texexec`, rather than `tex`, is used to compile the document. `texexec` is a shell script that do lots of things besides sending the source file to `tex`, such as post-process PDF files, run METAPOST to generate figures, etc. `texexec` can generate PDF files directly via pdfTEX, or, if called without `--pdf` option, traditional DVI files.

2.2 Extended Hello World

Now that we know how to typeset a simply line of text, we want more fancy stuff. First thing we need may be a title. Generally there are two ways. We can apply a bold face with large size on the title text, and make it middle-aligned, which can be done with

```
\midaligned{\tfc\bf A Dumb Title}
```

Example

A Dumb Title

The second method is claim the title text as a title

```
\title{Another Dumb Title}
```

Example

Another Dumb Title

The second result is sucks because we have not set it up. Put the following before `\starttext`

```
\setuphead
  [title]
  [textstyle=\bf,
    alternative=middle,
    after={\blank[4*big]}]
```

Recompile the source file and we get

Example

Another Dumb Title

So we have the two methods now. Which one should we use? The answer is the latter one, for the reason that it separates structure of our document from presentation. In ConTEXt, a `\title` is similar to `\section`, which means that we are going to begin to say somethings that share one thesis.

Besides this dumb title, maybe we need some different fonts and sizes in the main text. There are three kinds of fonts in TEX: roman, sans-serif

and `typewriter`, and 6 kinds of styles: normal, *italic*, **bold**, *slanted*, ***bold italic***, ***bold slanted***. All corresponding switches are shown in table 1.

	\rm	\ss	\tt
\tf	normal	sans-serif	typewriter
\it	*italic*	*italic ss*	*italic tt*
\bf	**bold**	**bold ss**	bold tt
\sl	*slanted*	*slanted ss*	*slanted tt*
\bi	***bold italic***	**bold italic ss**	bold italic tt
\bs	***bold slanted***	**bold slanted ss**	bold slanted tt

Table 1 Table of all fonts and commands

But wait! Seems we mentioned that structure and presentation should be separated? That's right. Generally, we use bold or italic fonts to emphasize something. ConTEXt provides `\em` switch to do this.

```
The {\em gang} is really important.
```

Example

The *gang* is really important.

We have had a paragraph of text for now, but how to start a new paragraph? In ConTEXt, there are three ways to indicate a new paragraph. We can either use a newline, or `\paragraph`, or `\par` to separate paragraphs, and for readability of the source file, the first one is recommenced.

Ok, we have learned a few markups that enable us to typeset a very short essay. In later sections, we shall discuss some more advanced topics.

3 Page Layout

The mechanism ConTEXt used to adjust page layout is quite different from plain-TEX and LATEX. We can either set page dimensions directly or use subpapers.

Some frequently used dimensions are shown in figure 1. However, in ConTEXt's official manual, the usage of these dimensions are rarely discussed. I will talk about some common adjustments.

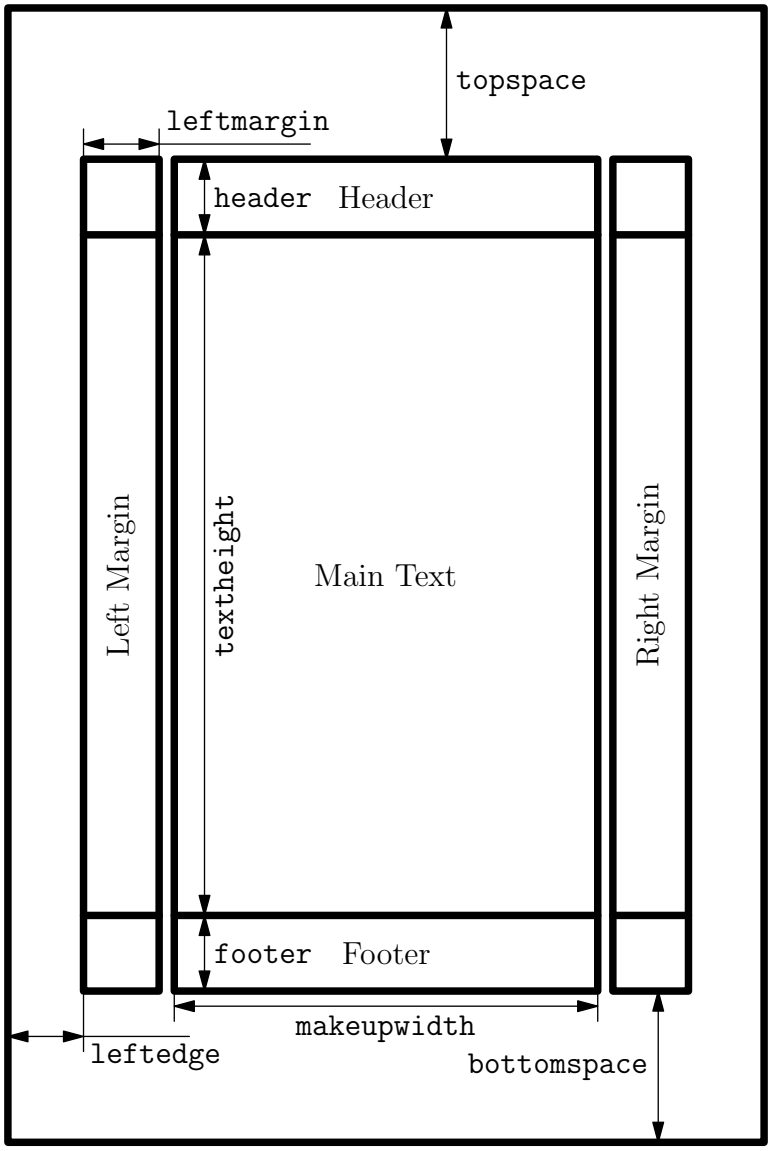


Figure 1 Some page dimensions

3.1 Reduce The Height

To reduce the text height, we can either add white spaces above headers and beneath footers, or make headers and footers bigger. To add white spaces, we shall set the `topspace`.

```
\setuplayout
  [height=middle,
    topspace=2cm]
```

`height=middle` means that when we adjust one of `topspace` and `bottomspace`, the other will be automatically set to ensure that the main text is vertically in the middle.

3.2 Reduce The Width

Strangely enough, I have not found any variables that can reduce the width while keep the text in the middle. If we directly reduce the value of `width`, the whole page will be left-adjusted, and there will be a area of white space on the right of the page. As an alternative, we can reduce `textwidth`, but notice that this value means the width of *one* column, if we use two columns, the total width of text will be a little more that twice of `textwidth`. What is worse, while the main text becomes thinner, the margins will remain there, next to the borders of the papers.

For now, I use a walk-around of this problem that place a thinner paper on our standard A4 paper.

```
\definepapersize[A4IN][width=17cm,height=25cm]
\setuppapersize[A4IN][A4]
```

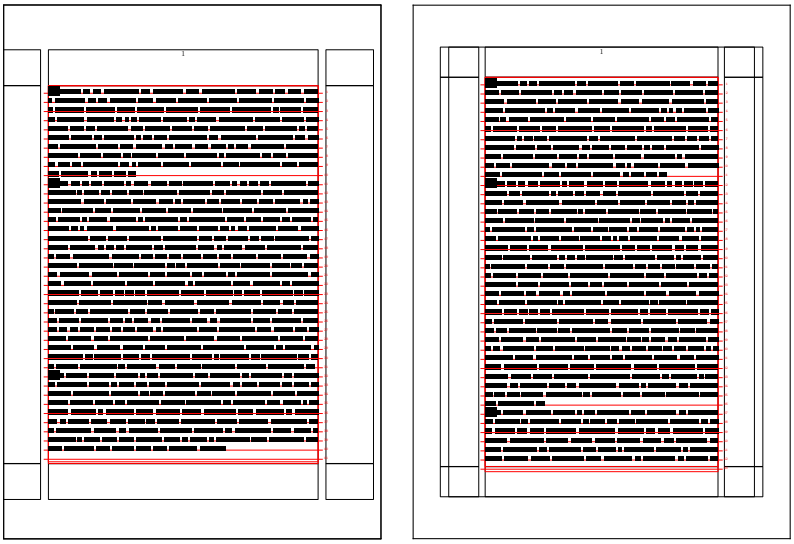
The standard A4 paper is 210 mm wide and 297 mm high. And I use a smaller paper with width of 17 cm and height of 25 cm to reduce both height and width of text. To make sure that main text will be at the center, setup the layout

```
\setuplayout
  [width=middle,
    height=middle,
    location=middle,
    topspace=0mm]
```

`location=middle` means that ConTEXt puts the smaller paper at the center of A4 paper, and `topspace=0mm` force `makeupheight` (= `textheight`+ `footer` + `header`) be equal to the height of the smaller paper. Figure **2** shows the result. In fact, we can use subpages to achieve any text size we want.

4 Typeset Plain Text

Basically, when typesetting plain text, ConTEXt and LATEX share the same rules with TEX because they are both macro-packages of TEX. We



No setup Width reduced

Figure 2 Use subpage to reduce the width of page.

just type the text as we usually do, and separate two paragraphs with an empty line. Also, we have known how to use some fancy styles on out text from section **2.2**, "**Extended Hello World**". In this section, we shall discuss some more advanced techniques.

4.1 Alignment

In most cases, we setup some environments to align text automatically for us. For example, in section **2.2**, "**Extended Hello World**", we showed how to setup titles to be middle-aligned. But in some minor situations, we still want to manually align text ourselves. To manually align text, we can use `\leftaligned`, `\rightaligned` and `\midaligned` command as section **2.2** showed, or, to align a large block of text instead of one line, use the `alignment` environment.

```
\startalignment[middle]
  \startlines
    Then am I,
    A happy fly,
    If I live,
    Or if I die.
  \stoplines
\stopalignment
```

Example

Then am I,
A happy fly,
If I live,
Or if I die.

The option passed to `startalignment` can be `width`, `left`, `right`, `middle`, `inner`, `outer`, `wide`, `broad`, `height`, `bottom`, `line`, `reset`, `hanging`, `nothanging`, `hyphenated` and `nothyphenated`. See what the official manual says about them:

> "The keys `left`, `middle` and `right`, `inner` and `outer` apply to horizontal alignment and bottom, `height` and `line` to vertical alignment.
>
> The key `right` results in the text being typeset ragged right. The keyword `broad` can be combined with `left`, `middle` and `right` which results in somewhat more rough alignments.
>
> The option `line` lets the last line touch the bottom of the page while `height` aligns the baseline to the bottom."

4.2 New Lines

We have known that ConTEXt ignore single new lines in a paragraph. In case we really what to typeset new lines, ConTEXt provide several commands. The most simple method is to use a `\crlf` in the position we want to start a new line.

```
We do not need new lines.\crlf We surly don't.
```

Example

We do not need new lines.
We surly don't.

Also, to typeset a paragraph contains multiple new lines, the `lines` environment can be used. Every new line contained in this environment in the source file will be reserved.

```
\startlines
  We do not need a new line here,
  and here,
  and everywhere.
\stoplines
```

Example

We do not need a new line here,
and here,
and everywhere.

Compare these two results and we can find the difference between `lines` environment and `\crlf` that the paragraph typeset using `\crlf` is remained a literal paragraph, while that using `lines` environment becomes a sort of list which can be told from the fact that there is a indention at the beginning of the first example, while the second one has no indention.

In fact, we can setup `lines` to indent all lines, or even lines, or odd lines using the key `indenting`.

```
\setuplines[indenting=odd]
\startlines
  Stranger's passing in the street,
  By chance two separate glances meet.
  I am you and what I see
  Is me.
\stoplines
```

Example

Stranger's passing in the street,
By chance two separate glances meet.
I am you and what I see
Is me.

Please consult the official manual, page 73 for details.

4.3 Verbatim Text

Sometimes we need to typeset a block of text to exactly how it is in the source file. We can use the `typing` environment to do this, or use `\type` command to typeset in-line verbatim text. Also, we can add some fancy highlighting to typeset some specific language source including METAPOST (`MP`), METAFONT (?), PERL (`PL`), JavaScript (`JS`), SQL (`SQL`?) and TEX/ConTEXt (`TEX`) by using the respective environment. For example, we use `TEX` environment to typeset TEX/ConTEXt source code.

```
\startTEX
{\bf A fat man.}
\stopTEX
```

Example

```
{\bf A fat man.}
```

4.4 Footnotes

Footnotes are typeset using `\footnote` command.

```
Sometimes we need to typeset a block\footnote{A block is different
from a paragraph} of text to exactly how\footnote{How, how,
and how?} it is in the source file. We can use the typing environment
to do this, or use \type{\type} command to typeset in-line
verbatim text.
```

Example

Sometimes we need to typeset a block[1] of text to exactly how[2] it is in the source file. We can use the typing environment to do this, or use `\type` command to typeset in-line verbatim text.

[1] A block is different from a paragraph
[2] How, how, and how?

4.5 Columns

We can typeset text in multiple columns using the `columns` environment.

```
\startcolumns
Maxima (pronounced m\ae xim\rotate{\hbox{e}}) is a large computer
program designed for the manipulation of algebraic expressions.
 You can use Maxima for manipulation of algebraic expressions
involving constants, variables, and functions.
\stopcolumns
```

Example

Maxima (pronounced mæximə) is a large computer program designed for the manipulation of algebraic expressions. You can use Maxima for manipulation of algebraic expressions involving constants, variables, and functions.

In addtion, we can pass an option `n=`n to the `columns` environment in which n is the number of columns.

There seems to be a bug that columns defined using this method have quite a big white space before the content.

Furthermore, we can fine control where to start the new column using the `paragraph` environment. Firstly we have to define a new paragraph:

```
\defineparagraphs[DumbCol][n=2]
```

Then we use this new environment and an automatically-defined command `\DumbCol` to control the columns:

```
\startDumbCol
```

```
Maxima (pronounced m\ae xim\rotate{\hbox{e}}) is a large computer
program \DumbCol designed for the manipulation of algebraic
expressions.  You can use Maxima for manipulation of algebraic
expressions involving constants, variables, and functions.
\stopDumbCol
```

Example

Maxima (pronounced mæximə) is a large computer program designed for the manipulation of algebraic expressions. You can use Maxima for manipulation of algebraic expressions involving constants, variables, and functions.

4.6 Framed Text and Lines

4.6.1 Frames

Sometimes we want to emphasize something by frame it, which can be done by `\framed` and `inframed` command.

```
And next unto them repaired
Rephaiah \framed{the son of
Hur}, \inframed{the ruler} of
the half part of Jerusalem.
```

Example

And next unto them repaired Rephaiah the son of Hur, the ruler of the half part of Jerusalem.

And there is a `framedtext` environment to frame a large block of text if one does not want to use `\framed`.

```
\startframedtext
And next unto them repaired
Rephaiah the son of Hur, the
ruler of the
half part of Jerusalem.
\stopframedtext
```

Example

And next unto them repaired Rephaiah the son of Hur, the ruler of the half part of Jerusalem.

Notice the white space around the text and before, after the frame.

4.6.2 Lines

We can draw a horizontal straight line using `\thinrules` command.

```
Oops \thinrules[n=3]\ the
line ends here.
```

Example

Oops

the line ends here.

In the code shown above, The `\␣` after `[n=3]` is to make sure the existence of the white space between the end of the line and the text "the line ends here".

4.7 Lists

Lists in ConTEXt are far more complicated than in LATEX, and mean much more than just literal lists. A general way to produce a list is to use the `itemize` environment.

```
\startitemize
\item Rocket Launcher
\item Plasma Gun
\item BFG
\stopitemize
```

Example

- Rocket Launcher
- Plasma Gun
- BFG

We can give `itemize` environment an option indicating the symbol prefixing every item.

```
\startitemize[g]
\item Rocket Launcher
\item Plasma Gun
\item BFG
\stopitemize
```

Example

α. Rocket Launcher
β. Plasma Gun
γ. BFG

All the prefixes available are shown in table **2**.

option	prefix	option	prefix
`n`	1,2,3	`1`	dot
`a`	a,b,c	`2`	dash
`A`	A,B,C	`3`	star
`KA`	A,B,C	`4`	triangle
`r`	i,ii,iii	`5`	circle
`R`	I,II,III	`6`	big circle
`KR`	I,II,III	`7`	bigger circle
`m`	1,2,3	`8`	square
`g`	α,β,γ		
`G`	A,B,Γ		

Table 2 Table of all list prefixes